

⑩ 日本国特許庁(JP)

⑪ 特許出願公開

⑫ 公開特許公報(A)

昭62-283496

⑬ Int.Cl.

G 11 C 17/00

識別記号

307

庁内整理番号

6549-5B

⑭ 公開 昭和62年(1987)12月9日

審査請求 未請求 発明の数 1 (全9頁)

⑮ 発明の名称 プログラマブルリードオンリメモリの書き込み回数管理方式

⑯ 特 願 昭61-124731

⑰ 出 願 昭61(1986)5月31日

⑱ 発 明 者 仲 田 真 一 東京都大田区下丸子3丁目30番2号 キヤノン株式会社内  
⑲ 出 願 人 キヤノン株式会社 東京都大田区下丸子3丁目30番2号  
⑳ 代 理 人 弁理士 小林 将高

明 細 書

1. 発明の名称

プログラマブルリードオンリメモリの書き込み回数管理方式

2. 特許請求の範囲

記憶領域に書き込まれた情報を電気的に消去可能なプログラマブルリードオンリメモリにおいて、前記記憶領域を複数のブロックに分割し、各ブロック毎に書き込み回数を記憶し、この書き込み回数に応じて書き込み頻度の低いブロックを未使用ブロックの先頭から書き込むとともに、前記書き込み頻度の低いブロックを未使用ブロックの最後尾に接続させることを特徴とするプログラマブルリードオンリメモリの書き込み回数管理方式。

3. 発明の詳細な説明

(産業上の利用分野)

この発明は、電気的消去可能なプログラマブルリードオンリメモリの書き込み回数の管理方式に関するものである。

(従来技術)

従来のEEPROM(Electrical Erasable and Programmable ROM)は、容量も少なく、また書き込むために必要な外部回路が多かった。さらに、チップ内のすべてのデータを消去するモードしか有していなかった。最近では、容量も大きくなるとともに、外部回路も殆ど必要なくCPUのアドレスバス、データバスに接続できるようになり、またEEPROM内の1バイトのデータのみ消去も可能となってきた。以上の改良により、使用目的によっては、従来のランダムアクセスメモリ(RAM)で構成していた機能の置換が可能となった。

例えば、従来の小型パソコン、日本語ワープロで作成したプログラムや文庫、外字等を保存しておくためにメモリカードと云うものがある。これは、必要となしにパソコン、日本語ワープロ等の本体に差し込んでプログラムや文庫を記憶させ、本体から引き抜いても、そのデータを記憶しているように、メモリカード内にはRAMと電池が搭

されていた。そこで、メモリカードをEEPROMで構成することにより、電池を無くすることができると思われた。

(発明が解決しようとする問題点)

ところが、EEPROMでは従来のRAMのように自由に何度も書き換えられない制約があり、すなわち、あらかじめ設定される書き込み回数を越えて、メモリカードへの書き込みを行うことにより、記憶しているはずのデータを消失させてしまう等の問題点があった。またEEPROMに書き込まれたデータのうち、頻繁に書き換えられるデータと書き換え頻度の少ないデータとが存在し、書き換え頻度の高いデータの書き換え回数が所定値を越えると、EEPROMへの書き換えが可能にも関わらず書き換え不能となる問題点があった。

この発明は、上記の問題点を解消するためになされたもので、EEPROMに書き込まれるデータの消失を防止するとともに、EEPROMへの書き込み回数を平均化させるとともに、EEPROM

OM上の書き換え頻度を平均化して、EEPROMへの書き換え寿命を延命できるプログラブルリードオンリメモリの書き込み回数管理方式を得ることを目的とする。

(問題点を解決するための手段)

この発明に係るプログラブルリードオンリメモリの書き込み回数管理方式は、記憶領域を複数のブロックに分割し、各ブロック毎に書き込み回数を記憶し、この書き込み回数に応じて書き込み頻度の低いブロックを未使用ブロックの先頭から書き込むとともに、前記書き込み頻度の低いブロックを未使用ブロックの最後尾に接続させる。

(作用)

この発明においては、記憶領域の各ブロック毎の書き込み回数を記憶しておき、この書き込み回数に応じて書き込み頻度の低いブロックを未使用ブロックの先頭から書き込ませるとともに、書き込み頻度の低いブロックを未使用のブロックの最後尾に接続させる。

(実施例)

第1図(a)はこの発明の一実施例を示すプログラブルリードオンリメモリへの書き込み回数管理方式を説明する模式図であり、1はEEPROMで、例えば書き込み容量が32768バイト×8ビットで、書き込み回数が1万回に設定してある。EEPROM1は、ポインタブロック1aおよび予備ポインタブロックSPB1~SPB50より構成される。ポインタブロック1aは4アドレス(各1バイト)で構成され、「0~1」番地の2バイトで、書き換え回数WCNT、例えば「138811」を記憶している。またポインタブロック1aの「2」番地の1バイトは、ディレトリDB、例えば「0111」を記憶している。さらに、ポインタブロック1aの「3」番地の1バイトは、未使用のスタートブロック番号OSB、例えば「3311」を記憶している。またポインタブロック1aの「4」番地の1バイトは、未使用のエンドブロック番号OEB、例えば「8A11」を記憶している。

第1図(b)はこの発明の装置構成の一例を説

明するブロック図であり、11はCPUで、ROM11a、RAM11bを有し、ROM11aに格納された第7図、第8図に示すフローに基いたプログラムに応じて各部を制御する。12は入力手段で、データ書き込み装置13にセットされるEEPROM1へのデータ書き込みおよびデータ消去を指示する。なお、CPU11にはデータの伝送を行うアキュムレータACC、BCCを有している。

第2図は第1図(a)に示すEEPROM1の構造を示す模式図であり、21はブロック番号であり、例えば127個のブロックBLOCK1~BLOCK127に分割されている。各ブロックは、例えば256バイトで構成され、先頭の2バイトで、そのブロックが更新された回数、すなわち、経過する更新回数が記憶されている。次に続く253バイトは記憶データDATAが記憶されており、最後の1バイトは、記憶データDATAがこのブロックに留まるか、または他のブロックに及ぶかどうかを示す最終ブロックエリアCBが

あり、他のブロックに記憶データDATAが及ぶ場合は、最終ブロックエリアCBには最終するブロック番号が記憶され、他のブロックに記憶データDATAが及ばない場合は、最終ブロックエリアCBには「FF<sub>16</sub>」が記憶されている。

第3図は第2図に示す各ディレクトリブロック構造を説明する模式図であり、30は前記ディレクトリDBに指示されるディレクトリブロック、31は前記ディレクトリブロック30の更新カウンタで、例えば2バイトで構成される。32はファイル領域で、各ファイル名が12バイトで記憶される。33はスタートブロック番号エリア(SB)で、例えば1バイトで構成され、ファイルのスタートブロック番号が記憶されている。34はエンドブロック番号エリア(EB)で、例えば1バイトで構成され、ファイルのエンドブロック番号が記憶されている。35はチェーンブロックエリア(CB)で、ディレクトリブロック30に最終するディレクトリブロックの有無を記憶する。例えばチェーンブロックエリア35が「FF<sub>16</sub>」

となる。なお、ディレクトリブロック30は、例えば18個のファイル領域32で構成される。

次に第1図(a)および第3図を参照しながらEEPROM1の構造について説明する。

第1図(a)に示すようにポインタブロック1aの書き換え回数WCNTに、例えば「1388<sub>16</sub>」が記憶されているとすると、5000回の更新が行われたことを示し、またディレクトリDBには「01<sub>16</sub>」が記憶されているので、ディレクトリDBに指示されるディレクトリブロック30のブロック番号が「1」で、そのディレクトリブロック30の更新カウンタ31には、「142F<sub>16</sub>」が記憶されている。これは、このディレクトリブロック30を5167回更新したことを示し、ファイル領域32のファイル(File)1(ファイル名)はスタートブロック番号エリア33が「02<sub>16</sub>」で、エンドブロック番号エリア34が「05<sub>16</sub>」となっているため、ブロックBLOCK2から始まり、ブロックBLOCK5で終ることになる。またファイル領域32のファイ

ル2は、スタートブロック番号エリア33が「0A<sub>16</sub>」で、エンドブロック番号エリア34が「0F<sub>16</sub>」となっているため、ブロックBLOCK10から始まり、ブロックBLOCK15で終ることになる。さらに、ファイル領域32のファイル3(ファイル名)は、スタートブロック番号エリア33が「15<sub>16</sub>」で、エンドブロック番号エリア34が「18<sub>16</sub>」となっているため、ブロックBLOCK21から始まり、ブロックBLOCK24で終ることになる。またファイル領域33のファイル3の次に「FF<sub>16</sub>」が書かれているので、このファイル領域33はファイル3で終了していることになる。

第4図は未使用のEEPROM1の状態を説明する模式図であり、第1図(a)、第3図と同一のものには同じ符号を付している。

この図から分かるように、未使用のEEPROM1のポインタブロック1aの書き換え回数WCNTが「0001<sub>16</sub>」、ディレクトリDBが「01<sub>16</sub>」、未使用のスタートブロック番号OSBが

「02<sub>16</sub>」、未使用のエンドブロック番号OEBが「7A<sub>16</sub>」がそれぞれポインタブロック1aの0番地から4番地にそれぞれ記憶されている。これにより、ディレクトリDBに指示されるブロックBLOCK1を参照すると、更新カウンタ31に「0001<sub>16</sub>」が書き込まれているとともに、ファイル領域32のファイル1に「FF<sub>16</sub>」が書き込まれており、さらに、チェーンブロックエリア35に「FF<sub>16</sub>」が書き込まれており、EEPROM1が未使用状態であることを示している。

さらに、ポインタブロック1aのスタートブロック番号OSBおよびエンドブロック番号OEBには「02<sub>16</sub>」、「7F<sub>16</sub>」がそれぞれ書き込まれている。すなわち、ブロックBLOCK2~127には先頭の2バイトに「0001<sub>16</sub>」が書き込まれ、最終の1バイトに各最終のブロックの最終を示すチェーンブロックエリア35には、ブロックBLOCK2~126に対して「03~7F<sub>16</sub>」が書き込まれ、ブロックBLOCK127のチェーンブロックエリア35には「FF」が書

込されている。このように、各ブロックBLOCK 2～127は1つのチェーン構造となる。

次に第3図、第5図(a)、(b)を参照しながらEEPROM1への書き込み動作を説明する。

第5図(a)、(b)はEEPROM1への書き込み動作を説明する模式図であり、第1図(a)、第3図と同一のものには同じ符号を付している。なお、書き込み直前は、第3図に示す状態であったものとする。

まず、各ブロックBLOCKのファイル領域32の先頭が「00<sub>16</sub>」のところを探し当てる。第3図の場合は、ファイル2とファイル3との間に「00<sub>16</sub>」があり、そこにファイル1という名前を12バイトで書き込み、ポインタブロック1aの未使用ブロックのスタートブロック番号OSBを参照して、スタートブロック番号OSBの指示するブロックBLOCK、すなわち「57<sub>16</sub>」の先頭の2バイト情報、すなわち、更新カウンタ31を「1」インクリメントし、その加算値

が、例えば1万回を越えているようであれば、ファイル4のチェーンブロックエリア35が示すブロックBLOCKに対して同様の操作を行い、更新カウンタ31が1万回以下のブロックBLOCKを探し当てる。そのブロックBLOCKの番号をポインタブロック1aのスタートブロック番号OSBに書き込むとともに、ファイル4のデータをブロックBLOCK87(253バイト)に書き込み、ブロックBLOCK87に格納されるようであれば、ブロックBLOCK87のチェーンブロックエリア35の指示するブロックBLOCKの更新カウンタ31を「1」インクリメントして加算値が、例えば1万回を越えているかどうかを調べ、指示されるブロックBLOCKの更新カウンタ31が1万回を越えるようであれば、更新回数が1万回以下のブロックBLOCKを探し当てる。そのブロックBLOCKの番号を直前に書き込んだブロックBLOCKのチェーンブロックエリア35に書き込む。このようにして、データの書き込みが行われ、更新回数が1万回を越えるブロッ

クBLOCKが排除されて行く。そして、書き込みデータがなくなるまで同様の操作を行い、最後に書き込んだブロックBLOCKのチェーンブロックエリア35に記憶されていた内容を新しい未使用のスタートブロック番号OSBに書き換え、ポインタブロック1aの書き換え回数WCNTを「1」インクリメントして「1389<sub>16</sub>」となり、最後にデータを書き込んだブロックBLOCKのチェーンブロックエリア35を「FF<sub>16</sub>」にする。そして、ディレクトリブロック30の最終ブロック番号を記憶するエンドブロック番号エリア34に最後のデータを書き込んだブロックBLOCKの番号を書き込むとともに、更新カウンタ31を「1」インクリメントすると、第5図(b)に示されるように、更新カウンタ31が「1430<sub>16</sub>」となり、ファイル4のスタートブロック番号エリア33が「33<sub>16</sub>」で、エンドブロック番号エリア34が「37<sub>16</sub>」となる。

次に第5図(a)、(b)を参照しながらEEPROM1に書き込まれているファイル1の削除

動作について説明する。

ディレクトリブロック30となるブロックBLOCK1よりファイル1を探し、ファイル領域32の先頭の2バイトを「00<sub>16</sub>」とする。次いで、ディレクトリブロック30の更新カウンタ31を「1」インクリメントし、ファイル1のスタートブロック番号エリア33とエンドブロック番号エリア34のデータを参照して、ポインタブロック1aのエンドブロック番号OEBが指示するブロックのチェーンブロックエリア35の内容(削除直前までは「FF<sub>16</sub>」であった)をスタートブロック番号エリア33の内容に変更し、このブロックの更新カウンタ31を「1」インクリメントする。すなわち、未使用ブロックの最後に今削除したファイル4を接続するわけである。このようにして、更新カウンタ31を進めながら何度もファイルの更新、削除を実行して行くうちに、更新カウンタ31が1万回に接近する。

次に更新カウンタ31が1万回に到達した場合のアクセス処理について説明する。

まず、ポインタブロック1aのスタートブロック番号OSBの内容が示しているブロックBLOCKのチェーンブロックエリア35の内容を新規のスタートブロック番号OSBとする。次いで、このブロック直前のディレクトリブロック30の更新カウンタ31の情報以外の内容を転送する。そして、ポインタブロック1aのディレクトリDBに新規のディレクトリブロック番号を書き込み、ポインタブロック1aの書き換え回数WCNTおよび更新カウンタ31を「1」インクリメントする。

一方、ポインタブロック1aの書き換え回数WCNTは1万回を越えた場合は、予備ポインタブロックSPB1～SPB50のうち一番近い予備ポインタブロックへ書き換え回数WCNTの情報以外のデータを転送し、新規のポインタブロックの書き換え回数WCNT(000016)を「1」インクリメントして「000116」に設定する。この場合、破棄されたポインタブロック1aの書き換え回数WCNTは1万回以上となり、新のポ

インタブロック1aの書き換え回数WCNTは1万回以下となる。このようにして、ディレクトリブロック30およびポインタブロック1aの書き込み削除を管理する。また削除されたファイルが使用していたブロックは未使用ブロックの一番最後に回される。これは、未使用ブロックの使用回数を平均化するためである。しかし、使用されているファイルが更新されずにずっとそのままであると、そのファイルが使用しているブロックは更新回数そのまま変化しない。例えば、最初に作成されたファイルがそのままずっと登録されたまま残っていると、他のブロックは更新回数が5000回以上なのに、このファイルだけは2回というようなアンバランスが生じる。そこで、EEPROM1の使用状態を平均化するための補正処理を行う。

補正処理起動条件は下記(a)、(b)の場合においてである。

(a) ポインタブロック1aの書き換え回数WCNTの値が256の整数倍になった時点。

(b) ファイルを構成するブロックの更新カウンタ31の平均値が一番低い値と、未使用ブロックの更新カウンタ31の平均値との差が256を越えた時点。

すなわち、新規のファイルを作成したり、削除したりした後、ポインタブロック1aの書き換え回数WCNTを見て、ちょうど256の整数倍、2バイトの16進数の下桁が「0016」になった時点で、ディレクトリブロック30に登録されている順にファイルを検査して行く。そして、ファイルを構成するブロックの更新カウンタ31を加算し、さらに構成するブロック数で除して更新平均値を算出する。次いで、次のファイルに対しても同様の更新平均値を出して比較し、低いものをその比較対照として残し、次のファイルの更新平均値と比較して行き、一番低い更新回数で構成されるファイルを探し出す。そして、未使用のブロックの更新回数の平均を計算し、平均更新回数の一番低いファイルとの差を算出する。

次に第6図(a)～(c)を参照しながら補正

処理動作について説明する。

第6図(a)～(c)はこの発明による補正動作を説明する模式図であり、これらの図において、41はファイルで、ブロックBLOCK5～7で構成され、平均更新回数が最も低いものである。42は未使用ブロック群で、ポインタブロック1aの未使用スタートブロック番号OSBで指示される。未使用ブロック群42は、ブロックBLOCK50、10、11、18、55、80、81が1つのチェーン構造となっている。

まず、ファイル41のスタートブロックを未使用ブロック群42のブロックBLOCK81の最後に接続させるため、同図(a)に示すようにブロックBLOCK81のチェーンブロックエリア35の内容が「FF16」から同図(b)に示すように、チェーンブロックエリア35の内容が「0516」、すなわち、ファイル41のブロックBLOCK5を指示させ、さらに、ファイル41のブロックBLOCK5～7(ファイル41のチェーンブロックエリア35の内容が「FF16」になる

まで)を未使用ブロック群42のブロックBLOCK 81の後に接続させ、同図(c)に示すように、ディレクトリブロック30のファイル11のスタートポイントをブロックBLOCK 50にするとともに、終了ポイントをブロックBLOCK 11に変更する。次いで、未使用ブロック群42の未使用のスタートブロック番号OSBを「121」、未使用のエンドブロック番号OEBを「0B1」にする。これにより、登録されたファイルを構成するブロックに対しても再度使用可能となり、EEPROM1全体のブロックが平均的に使用、更新されることになる。第7図は第1図(a)に示したEEPROM1のデータ書き込み制御動作を説明するためのフローチャートである。なお、(1)～(18)は各ステップを示す。

まず、ディレクトリブロック30の空エリアを探して、新規のファイル名を書き込む(1)。次いで、未使用のスタートブロック番号OSBをCPU11のアクムレータACCに記憶させる(2)。アクムレータACCが指示するブロック

の書き換え回数WCNTを+1更新する(3)。ここで、書き換え回数WCNTが10000を超えたかどうかを判断し(4)、YESならばアクムレータACCの指示するブロックの最終ブロックエリアCBをアクムレータACCに記憶し(5)、ステップ(3)に戻り、NOならばディレクトリブロック30のスタートブロック番号エリア(SB)33にアクムレータACCの内容を書き込む(6)。次いで、アクムレータACCが指示するブロックのデータエリアにデータを書き込む(7)。ここで、書き込みデータがアクムレータACCが指示するブロックの容量が235バイトを超えるかどうかを判断し(8)、YESならばアクムレータACCが指示するブロックの最終ブロックエリアCBをアクムレータBCCに記憶させる(9)。次いで、アクムレータBCCが指示するブロックの書き換え回数WCNTを+1更新する(10)。次いで、書き換え回数WCNTが10000を超えたかどうかを判断し(11)、YESならばアクムレータBCCの指示するブロック

の最終ブロックエリアCBを記憶させ(12)、ステップ(10)に戻り、NOならばアクムレータACCが指示するブロックの最終ブロックエリアCBにアクムレータBCCの内容を書き込み(13)、ステップ(7)に戻る。

一方、ステップ(8)の判断でNOの場合は、アクムレータACCが指示する最終ブロックエリアCBを未使用のスタートブロック番号OSBに書き込む(14)。次いで、ポインタブロック1aの書き換え回数WCNTを+1更新する(15)。次いで、アクムレータACCが指示するブロックの最終ブロックエリアCBへ「FF」を書き込む(16)。そして、ディレクトリブロック30の新ファイル位置のエンドブロック番号エリア34へアクムレータACCの内容を書き込む(17)。次いで、ディレクトリブロック30の書き換え回数WCNTを更新する(18)。

第8図はこの発明による補正制御動作手順を説明するためのフローチャートである。なお、(1)～(7)は各ステップを示す。

ポインタブロック1aの書き換え回数WCNTが256の整数倍であるかどうかを判断し(1)、NOならばリターン(RETURN)し、YESならばディレクトリブロック30に登録された各ファイルを構成するブロックの更新カウンタ31の平均値を算出して、最も更新回数が少ないファイルを探し出す(2)。次いで、未使用のブロックの更新回数の平均値を算出する(3)。次いで、未使用ブロックの更新カウンタの平均値からファイルを構成するブロックの更新カウンタの平均値の最小値を減算し、さらに減算値から256を差し引いた値が正かどうかを判断し(4)、NOならばリターンし、YESならば未使用ブロックの最後尾に該当するファイルのヘッドを接続させる(5)。次いで、接続したファイルの内容を未使用ブロックへ伝送させ(6)、ディレクトリブロック30にある接続したファイルのスタートポイント、エンドポイントを変更し(7)、リターンする。

(発明の効果)

以上説明したように、この発明は記憶領域を複数に分割し、各ブロック毎に書き込み回数を記憶し、この書き込み回数に応じて書き込み頻度の低いブロックを未使用ブロックの先頭から書き込むとともに、書き込み頻度の低いブロックを未使用ブロックの最後尾に接続させるようにしたので、EEPROMに書き込まれるデータの消失を防止するとともに、EEPROMへの書き込み回数を平均化させることができる。またEEPROM上の各ブロックの書き換え頻度を平均化でき、書き換え寿命を大幅に延長できる優れた利点を有する。

#### 4. 図面の簡単な説明

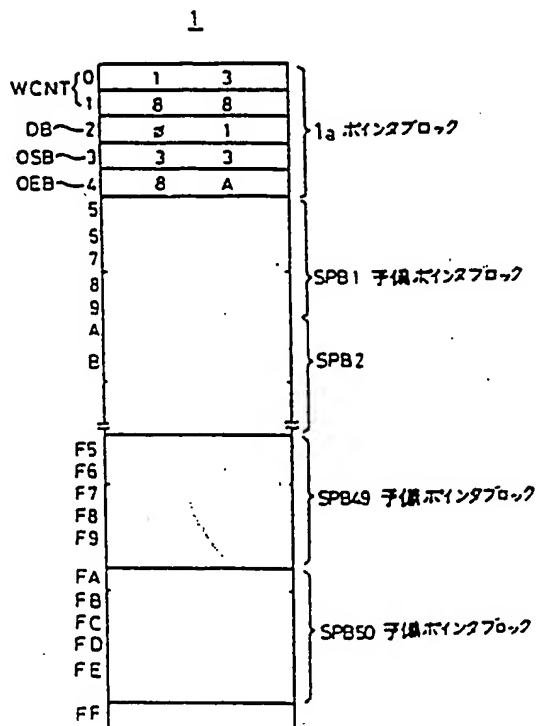
第1図(a)はこの発明の一実施例を示すプログラマブルリードオンリメモリへの書き込み回数管理方式を説明する模式図、第1図(b)はこの発明の装置構成を説明するためのブロック図、第2図は第1図(a)に示すEEPROMの構造を示す模式図、第3図は第2図に示す各ディレクトリブロック構造を説明する模式図、第4図は未使

用のEEPROMの状態を説明する模式図、第5図(a)、(b)はEEPROMへの書き込み動作を説明する模式図、第6図(a)~(c)はこの発明による補正処理動作を説明する模式図、第7図は第1図(a)に示したEEPROMのデータ書き込み制御動作を説明するためのフローチャート、第8図はこの発明による補正制御動作手順を説明するためのフローチャートである。

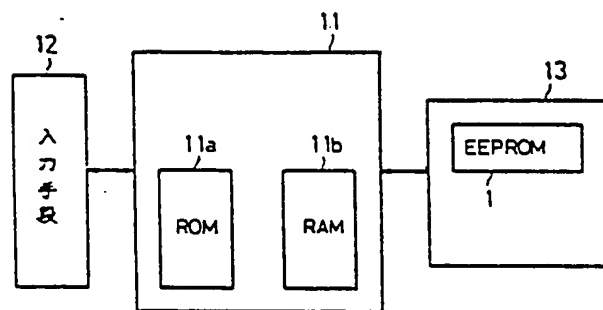
図中、1はEEPROM、1aはポインタブロック、21はブロック番号、30はディレクトリブロック、31は更新カウンタ、32はファイル領域、33はスタートブロック番号エリア、34はエンドブロック番号エリア、35はチェーンブロックエリア、41はファイル、42は未使用ブロック群である。

代理人 小林 将 高

第 1 図 (a)

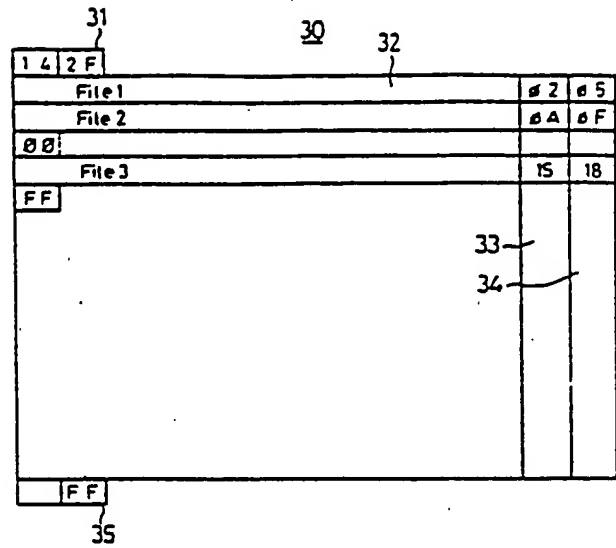
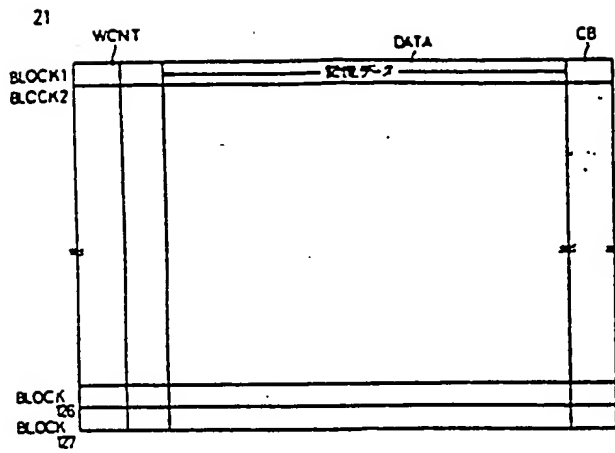


第 1 図 (b)



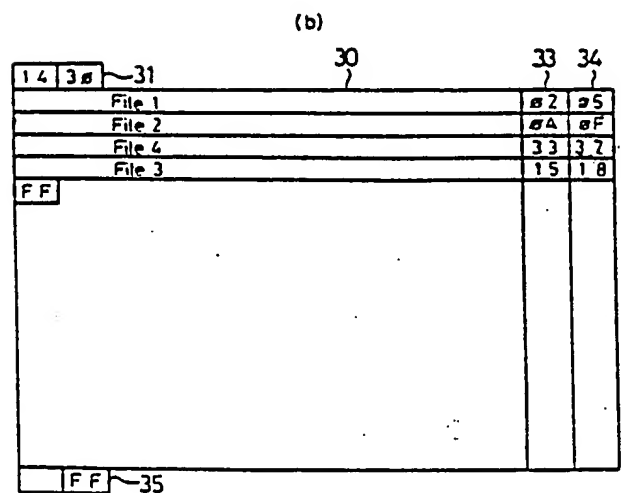
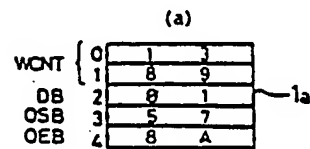
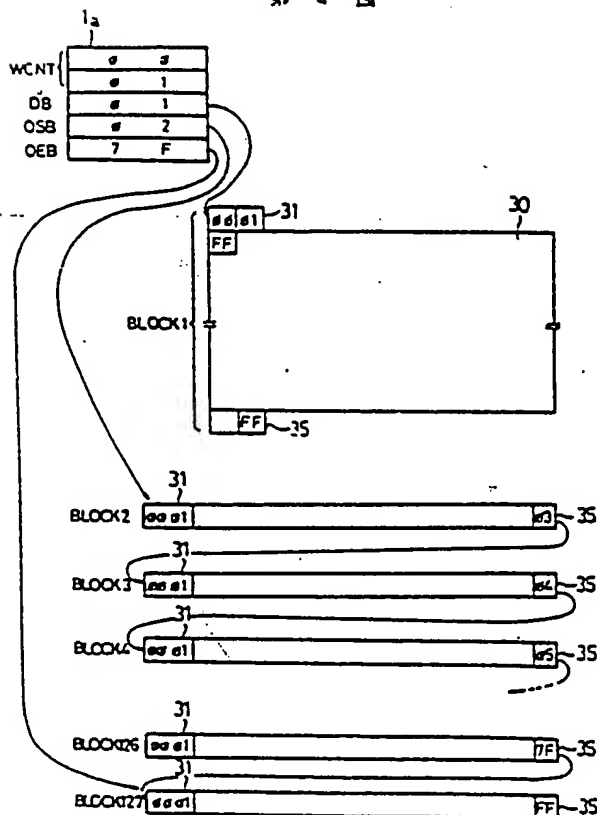
第 3 図

第 2 図



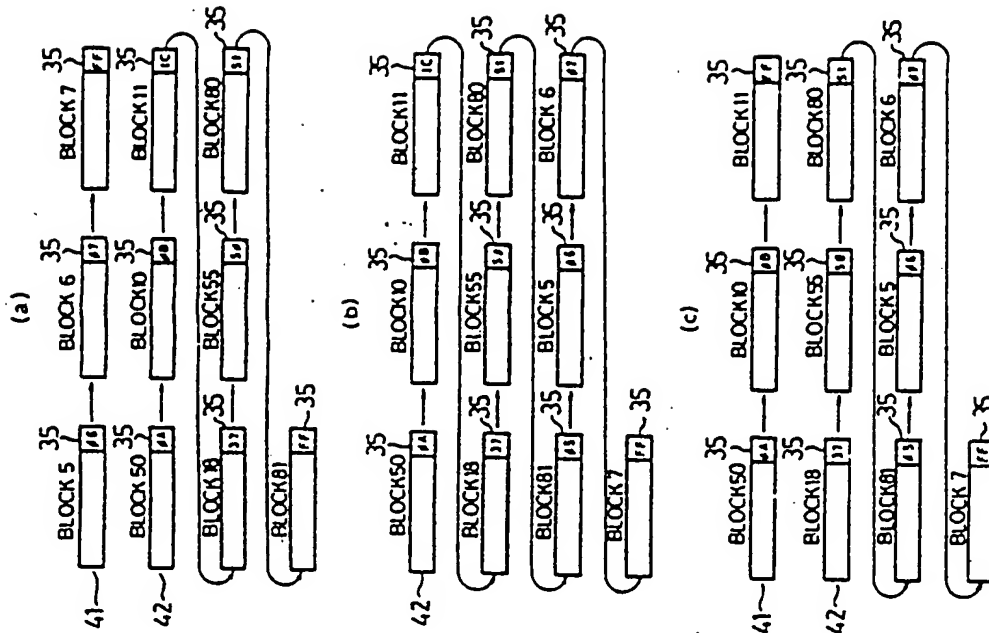
第 4 図

第 5 図

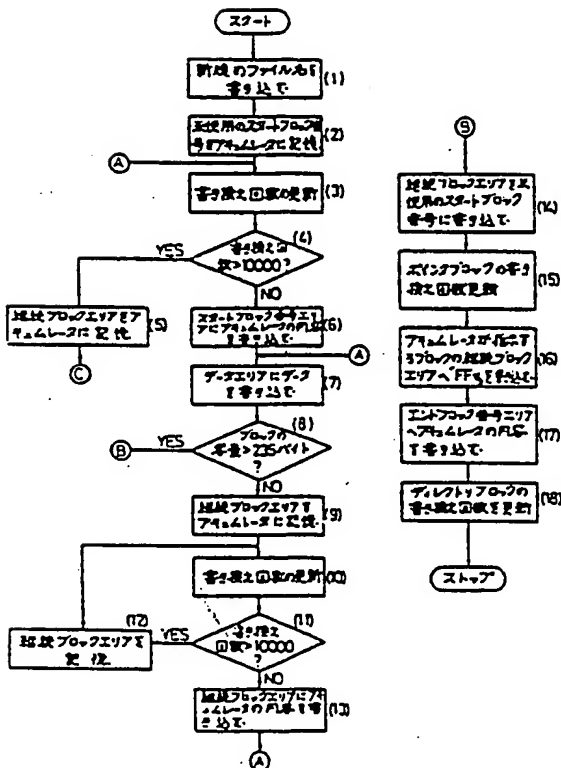




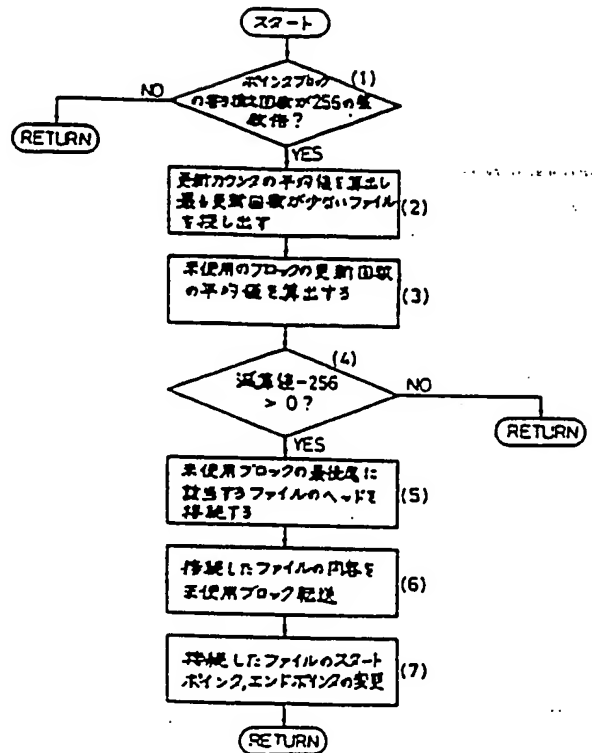
第 6 図



第 7 図



第 8 図



a: table 1. b: memory transistor. c: output

## AL

Diagram illustrating a memory layout structure:

	WONT	DATA	CS
BLOCK 1		6	
BLOCK 2			
BLOCK 3			

**a: storage data**

PROGRAMMABLE READ ONLY MEMORY (PROM) (17) 62-283497 (A) (43) 9.12.1987 (19) JP  
 (21) Appl. No. 61-124732 (22) 31.5.1986  
 (71) CANON INC (72) SHINICHI NAKADA  
 (51) Int. Cl. G11C17.00

Diagram illustrating a memory layout structure. The structure is a table with columns labeled 'WCHT', 'DATA', and 'CR'. The rows are labeled 'BLOCK 1', 'BLOCK 2', 'BLOCK 3', and 'BLOCK 4'. A handwritten '62' is present in the center of the 'DATA' column.

a: storage data